Object-Oriented Applications Development With NeXTstep

Object-Oriented Applications Development With NeXTstep

Executive Summary

Rapid development and deployment of mission-critical custom software applications are key to success in the 1990's. This paper describes the advantages of developing those applications using NeXTstep, the industry's most extensive object-oriented systems software. NeXTstep's advantages include its thorough object orientation and its integrated set of development tools. Programmers can develop applications five to ten times faster since they need to write less code. Benefits also include reduced learning time, improved maintainability and reliability, code reusability, increased programming flexibility, and the ability to manage more complex applications.

For users, NeXTstep applications can be developed and implemented more quickly. NeXTstep applications have professional-quality interfaces making them easier to use, and which integrate well with the productivity applications needed by users. This paper also describes the tools available to NeXTstep programmers, including NeXTstep's support for client-server computing and industry standards.

I. Introduction: Building Mission-Critical Custom Applications with NeXTstep

"The other workstation vendor to watch is NeXT. . . . Developers positively love it. Every NeXTstation comes with a complete set of development tools, and there is simply no better environment for building graphical applications. . . . People who are now using the NeXTs are nothing short of gaga over it, and their lust is justified."

– Byte, Outlook '92 issue, p. 164

Mission-critical custom applications are key to success in the '90's for organizations of all kinds. Custom applications are needed by virtually every large and medium-sized enterprise. But building custom applications is proving to be very difficult. The demand for graphical user interfaces has risen sharply, placing greater demands on developers. And even without the demand for friendlier applications, MIS groups are faced with a three-to seven-year backlog of new applications to build and old applications to maintain.

One response has been to port applications from mainframes to workstations. A related response has been to upgrade to machines with faster processors. But neither downsizing nor increasing the raw horsepower available to programmers has produced applications more rapidly or with appreciably more functionality than older mainframe software.

Why? While desktop computing has made great leaps in performance, doubling in power every two years, improvements in the way software is written have occurred at a snail's pace. The porting of corporate applications from mainframes to networks of workstations has not remedied the problem, since downsizing has often meant simply using old development tools and methods on faster workstations.

To improve productivity, programmers will have to modernize their development methods and tools. Software must become more maintainable and code more modular and reusable. But most importantly, programmers must use tools that require them to write less code.

Developers need a new method of writing software–and that new method is object-oriented programming (OOP). Developers need a rich and complete set of object-oriented tools–and that set of tools is NeXTstep.

II. NeXTstep

NeXTstep is object-oriented system software (OOSS) for workstations, designed to help organizations close the applications gap by making programmers more productive. NeXTstep improves programmer productivity in two ways. First, NeXTstep is a thoroughly object-oriented software architecture. Providing a rich set of prebuilt objects for common functionality, NeXTstep allows programmers to write less code. Second, NeXTstep provides a complete set of development tools which have been crafted to work together.

NeXTstep Is Thoroughly Object-Oriented

Object-oriented programming is the first significant revolution in software technology since the development of the graphical user interface. The entire architecture of NeXTstep is designed to support object orientation.

Objects are the key to understanding object orientation. They are self-contained chunks of code containing data and associated behavior (procedures). Objects send messages to each other requesting information and services. Together, these objects and the messages they send comprise a complete software application.

A software object represents something in the real world such as a user interface button or window, a data entry form, a business graphic, an employee record, or an information browser. Objects allow programmers to break a problem into small manageable components, modules of code, making it simpler for the programmer to design a logical structure. This process has two main benefits:

- Other engineers can design additional objects that implement features and treat the rest as a "black box" and
- Ramp-up time is shortened when a different engineer takes over maintenance of the software

Objects can be reused. Organizations can construct libraries, for example, of common analysis, forms, and report objects which can be used across many kinds of applications.

Object-oriented programming helps developers manage complex applications because they only need to understand the messages that objects send to each other to understand how a program works.

Sending a message to an object only requires requesting *what* needs to be done, without needing to understand *how* the task will actually be done. Object-oriented programs hide the details from everyone but the programmer of that object. Programmers focus on the architecture of the system: how the application is modularized into real-world objects offering specific kinds of functionality and how the application manages this functionality by passing messages between objects. Developers can operate on a higher level of abstraction and can ignore implementation details subject to revision as the software matures.

Being an object-oriented development platform, NeXTstep simplifies program development, makes cooperation among programmers easier and maintenance simpler, and gives the programmer a more comprehensive view of a program's architecture.

NeXTstep Provides A Complete Set of Tools

To its object-oriented programming language, Objective-C, NeXT has added object-oriented system software, tools for developing interfaces and managing objects, and a complete object framework for adding sophisticated functionality without requiring programmers to write additional code.

NeXTstep provides an extensive set of fundamental building blocks, an Application Kit of more than one hundred objects. This Kit reduces substantially the amount of code that needs to be written.

NeXTstep's Applications Kit supplies all of the features of a consistent user interface. NeXTstep makes all applications easier to create, enhancing programmer productivity. By providing complete tools to build and modify user interfaces, NeXTstep allows developers to focus on their application's unique features and encourages and enables them to develop custom applications with the look and feel of quality commercial software. And by furnishing user interface construction tools which encourage consistency, NeXT makes every application easier to use. By programming in NeXTstep, it is simple to create applications that are not only friendly but are also tailored to meet each organization's unique requirements.

With NeXTstep, programmers can reduce development time significantly, a valuable productivity gain for any business that relies on a mission-critical custom application.

> *"NeXT is really making a splash in launching SBC's [Swiss Bank Corporation] business in interest-rate derivatives. . . . Solo's team wrote the application in three months, which is half the time it would have taken on a Sun workstation, he says. 'I've never developed something so substantive in so little time.'"*
>
> – *Wall Street Computer Review*, Volume 9, No. 1 (1991), p. 46.

Recognized by the Industry

Given these advantages, it is no surprise that NeXTstep has received the Software Publishers' Association's *Fluegelman Award* for innovative software as well as *Computer Language* magazine's *Productivity Award* for interactive application development environments.

III. NeXTstep: A Consistent Philosophy

NeXTstep is the first desktop computing operating system environment designed for the developer of software applications.

NeXTstep was created to enable developers to reduce development time dramatically, by making common things easy to implement and difficult things possible and maintainable.

In creating NeXTstep, NeXT was guided by the following principles:

- Developers for the 1990's need an application development *architecture*, not a collection of poorly integrated tools that were not designed to work together.
- A complete development architecture must integrate programming languages, windowing and graphics systems, user interface toolkits and class libraries, database access tools, and program management tools.
- A development platform must provide a high level of standard functionality that all developers can depend on, functionality that will be common to most applications, leaving developers to write only the code which is unique to their application.
- Tools should be fully object-oriented to speed development and simplify maintenance, and all tools should use the same object description and extension language that the programming language uses.
- A development environment should employ the same graphics model for screen display and hardcopy output, and all applications should support multifont text and graphics standards.
- A development environment should encourage developers to provide high-quality, consistent graphical user interfaces, and interfaces which can be localized for different foreign languages without additional coding.
- The development environment should be well supported by the operating system. An object-oriented architecture should be complemented by an optimized messaging architecture which enables efficient object-oriented applications.
- Custom applications, because of hooks into the operating system, should integrate well with commercial productivity tools.

At NeXT we believe that these principles should be used to judge the adequacy of any development platform. Table 1 compares NeXTstep development tools and their capabilities to the tools available on another popular workstation development platform, Sun Microsystems.

IV. How NeXTstep Increases Programmer Productivity

> *"The strongest selling point for us is the NeXTstep development environment. It allows us to develop applications in roughly one-third the time (this is a blended figure over many applications). On a NeXT, you are encouraged, coddled, and brought quickly up what would otherwise be a difficult learning curve in making good use of OOP [object-oriented programming] technique. This is possible because object orientation isn't just a veneer on the programming environment. The heart and soul of the NeXT is object-oriented. This allowed our developers to become OOP experts with relatively little pain. The benefits are monumental. It is inconceivable to me that we would ever go back to the days of functional programming."*
>
> *– Hadar Pedhazur, Vice President, Equities Technology, Equity Derivative Products, UBS Securities, Inc.*

Faster Application Development

Programmers using NeXTstep's object-oriented development tools create software five to ten times faster than with traditional procedural programing languages. Why? Because NeXTstep programmers write less code.

	NeXT	Sun
Integrated Tools	Interface Builder manages all files, develops interfaces, extends class hierarchy, integrates with Objective-C.	Tools provided by different vendors; DevGuide only loosely coupled to C; DevGuide doesn't support any object-oriented language or toolkit.
Library of Objects for All Core Functionality	AppKit manages events, printing, faxing, and file management.	No applications framework for common core functionality.
Interface Development	Interface Builder (IB) manipulates live code, classes, and object messaging; custom objects can be displayed and manipulated by IB.	DevGuide lays out interfaces and generates static code; cannot display or manipulate custom objects.
Single Imaging Model	Display PostScript.	XLIB, NeWS, XGL, pixrects. No application printing model.
Standard Multimedia Data Formats	RTF, EPS, TIFF, sound; apps can cut and paste all of these formats.	No multifont text, sound, or toolkit support for imaging or data exchange.
Object-Oriented Tools	Optimized for Objective-C, can also use C++; consistent object framework across all tools: extension language common to AppKit and rest of code; run-time binding allows extensible objects and apps. IB encourages modularization of program.	Optimized for C; Native C environment for procedural, not object-oriented programming; C++ becoming available but is used procedurally with DevGuide. DevGuide creates monolithic non-modular code.
Consistent User Interfaces	NeXTstep provides complete dialogs in addition to windows and controls provided by other systems; makes it easy and natural to make consistent UI, hard to make inconsistent UI.	Motif, Open Look tools do not provide dialogs and panels; developers create own versions of common UI elements (e.g., print panels).
Development Tools Integrated With OS	Object-oriented system software; object architecture is built on top of Mach multithreading and messaging.	OS not designed with object-oriented development or deployment in mind.
Custom Apps Integrate With Commercial Apps / Interapplication Communication	All apps can be messaged and offer services to other apps; common message table architecture in all apps; apps can register ability to provide services with OS; apps can take advantage of other apps without knowing anything about their protocol or architecture.	ToolTalk messaging will be available to some apps. All applications must know protocol and architecture of other applications.

Table 1: NeXTstep vs. Sun Applications Development Environments

Being able to use any of the more than one hundred commonly used objects supplied by the Applications Kit (AppKit) allows developers to incorporate features such as spell-checking and multiple fonts without writing any additional code.

Reduced Learning Time

Unlike C++, Objective-C can be learned quickly. Programmers familiar with C can learn NeXTstep's Objective-C in a just a few hours because it is based on simple extensions to the C language. In addition, NeXT's integrated development tools includes online documentation stored in NeXT's Digital Librarian, and a run-time application inspector to aid mastery of the NeXT programming environment. The Application Kit has many well-formed examples of effective design and efficient code which help novice NeXTstep programmers learn object-oriented programming technique.

Improved Maintainability and Reliability

Traditional programming languages produce code that is difficult to maintain. Changes to one portion of the code may have unintended consequences which propagate throughout the application. Thus, maintaining software written in traditional languages is often as or more difficult as creating the application was originally.

Unlike traditional programming languages, object-oriented programming in Objective-C defines functionality at a high level of abstraction. To use a window object, for example, the programmer only needs to know what messages ("Close" or "Resize") to send that object–not how the window object actually works. This means that new objects can be substituted for less functional or less efficient ones without affecting the rest of the system. Object orientation protects the program from the errors common in the "spaghetti code" produced by traditional programming techniques, where everything is accessible by everything else. Because of modularity, bugs are located more quickly. And once fixed, bugs stay fixed, because programmers find and fix bugs rather than experimentally merely fixing the symptoms.

Increased Flexibility and Extensibility

The ability to modify objects easily (and optimize one portion of the application without affecting other parts) creates greater programming flexibility. Experimenting with new methods of handling certain functions in one object will not affect other objects. Changes can be made while a program is being developed without introducing delays.

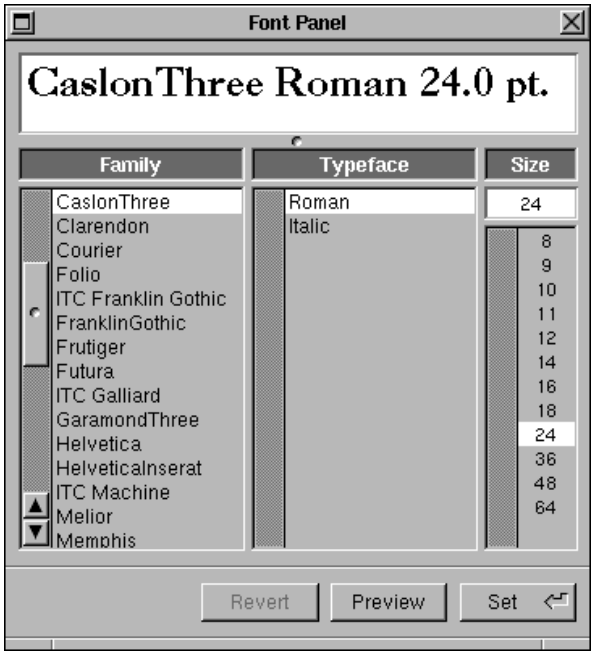

Figure 1. Reusable Code. This font panel from the AppKit is reusable across all NeXTstep applications.

All objects managed by NeXT's software management tool, Interface Builder, can be subclassed for added functionality. Interface Builder also encourages custom objects to be developed and shared, then added to the Interface Builder environment, where they can be graphically manipulated and managed. Palettes of many kinds of NeXTstep objects are commercially available from software companies today.

Reusable Code

NeXT's object-oriented programming environment encourages the development of many small functional modules (objects), each of which accomplishes a clearly defined task. By encouraging the development of these small functional objects, NeXT's object orientation adds to the overall reliability of development projects, since the objects have already been proven. (See Figure 1.)

Beyond modularity, reusability of code is promoted through *subclassing* and *inheritance*. Using subclassing, programmers can modify an object used by many groups without rewriting the object from scratch. By creating a special subclass of an

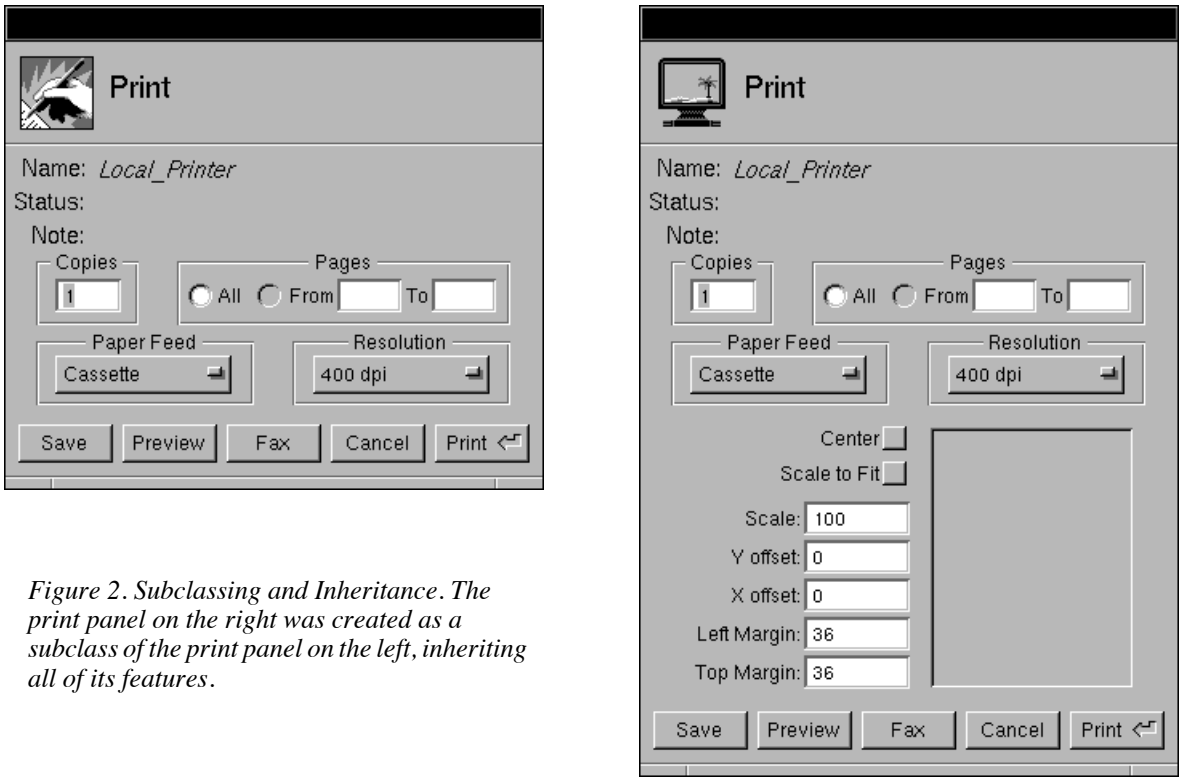

Figure 2. Subclassing and Inheritance. The print panel on the right was created as a subclass of the print panel on the left, inheriting all of its features.

expense form object which is common to an entire organization, a departmental approval process that is unique to one workgroup can be created by adding only a few lines of code, since all the common attributes will be inherited from the original object. (See Figure 2.)

Later, as common objects are modified and updated, all applications relying on them will automatically inherit the updates, modifications, and improvements. In this way, NeXTstep applications share fax and print panels that are written only once, but are used by virtually all applications and which can be updated without changing the applications themselves. Because NeXTstep encourage the creation of extensible and reusable programs, developers write fewer lines of code.

Manage Greater Complexity

By tracking the messages moving between objects, developers can track an application's development more efficiently. The development team works at a higher, more abstract level, caring less about how given objects work internally, concentrating instead on how the program is modularized into objects providing key kinds of functionality, and how objects communicate with each other.

V. NeXTstep: A Complete Object-Oriented Development Solution

NeXT is the first UNIX workstation vendor to provide and bundle a rich set of object-oriented software development tools with its computers. Unlike other UNIX workstation environments, NeXT includes *all* the development tools–editors, languages, class libraries, class browsers, object inspectors, a windowing system, and a unified imaging model– needed to develop sophisticated, graphical, and truly object-oriented applications.

All of NeXTstep's tools are designed to work together to provide a complete, integrated development solution. With NeXTstep, a development team is not burdened with poorly and only partially integrated class libraries, windowing systems, and programming languages which are maintained by different vendors, released at different times, and designed for different tasks. With NeXTstep, all the tools work together–optimized and seamless.

Interface Builder. Interface Builder is NeXTstep's primary tool for graphical interface development and project management. Studies have shown that as much as 90% of development time is devoted to constructing the user interface. Interface Builder

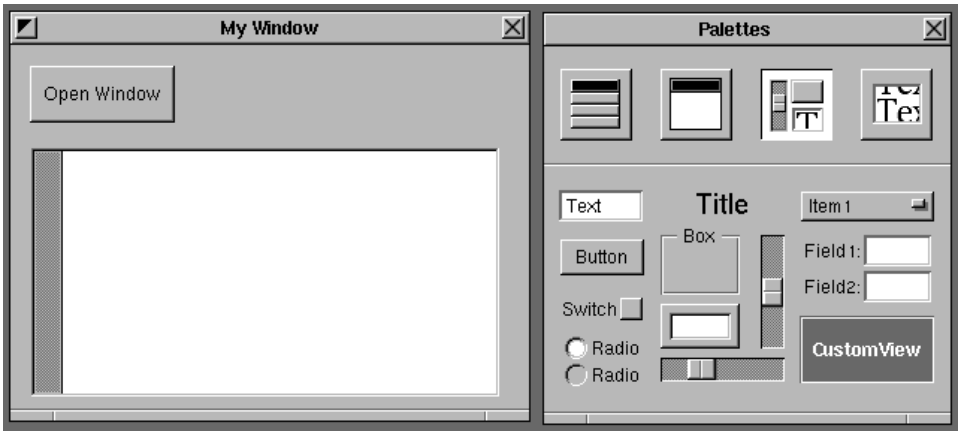

Figure 3. Interface Builder. Palettes of 'live' objects (right) can be dragged to windows (left) to create applications.

improves programmer productivity with its complete environment for laying out, constructing and testing user interfaces. (See Figures 3 and 4.)

Even if Interface Builder were no more than the most advanced interface and layout development tool on the market, it would have justly earned its breakthrough reputation on that strength alone. But Interface Builder is far more than a user interface tool. It provides powerful software engineering tools to manage object-oriented software projects. Interface Builder is:

- A complete object-oriented user interface development, layout, prototyping, and testing tool
- An object editor which manages the *interface* between all objects in the program; using Interface Builder, the programmer defines the protocol between objects (the messages they send each other) whether they are user interface objects or any other kinds of objects
- A software management tool, managing all components of a software project, including icons, C interface files, source code modules, and sound and image files; Interface Builder provides comprehensive "make" facilities for program construction and compilation
- A tool that encourages modularity because of the ease with which a program can be split into separate modules
- A localization tool which allows developers to lay out foreign language versions of their interface and enables an application to support multiple language interfaces, thus giving users the ability to run the application in the language of their choice
- A class hierarchy manager for NeXTstep classes and classes of objects added by developers, and a tool for subclassing and accessing class definitions
- An object and palette manager, allowing palettes of custom objects to be manipulated in the same way that NeXTstep Application Kit objects are displayed and manipulated

Unlike Sun, X/Motif, and Windows screen painters (which manipulate little more than static bitmap representations of widgets), Interface Builder brings live software objects into an application. These live objects can be interlinked and their behavior tested, and the entire interface can be revised in minutes. New objects can be brought into Interface Builder on custom palettes and then manipulated, shared, inspected, and used like any other NeXTstep objects.

Interface Builder involves users in software development projects. Interface Builder allows users to test and comment on the user interface through its interactive test mode. By providing painless refinement of the user interface early in the development cycle, Interface Builder helps ensures user participation and satisfaction.

Objective-C: A Rich Object-Oriented Language. NeXTstep's extensive object-oriented programming tools are based on Objective-C, an extension of standard ANSI C. Objective-C encourages developers to modularize their code, resulting in programs that are simpler to write and maintain. Objective-C combines the efficiency and familiar-

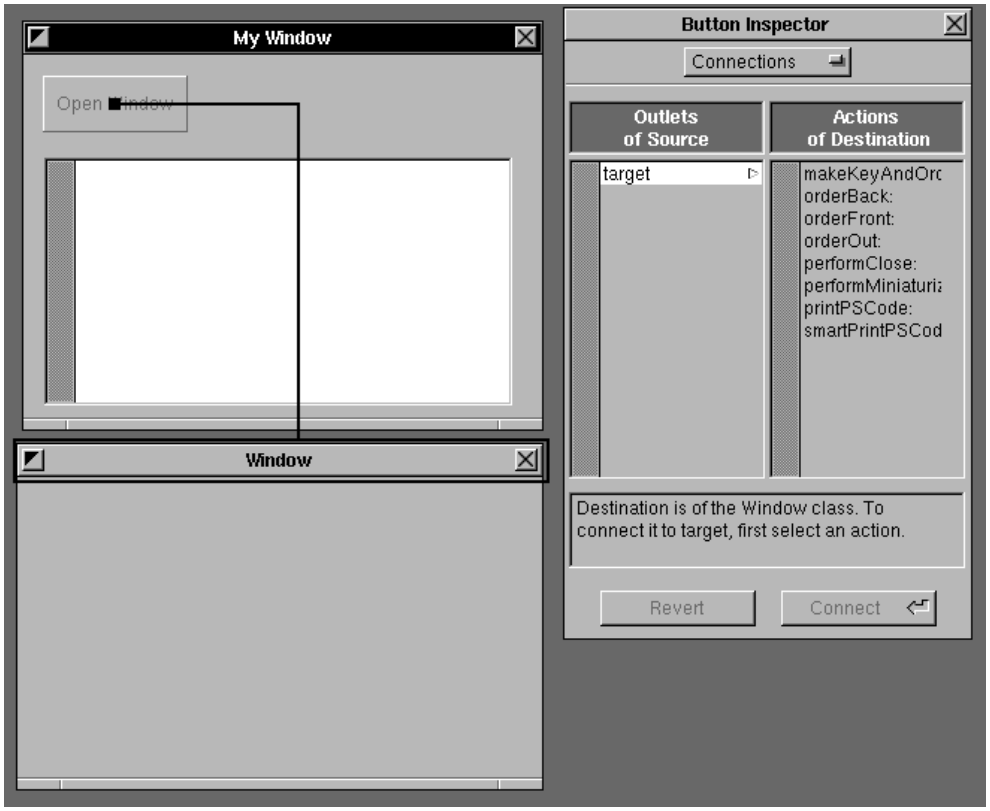

Figure 4. Interface Builder. Interface Builder is used to manage object messaging. The "Open Window" button in the top window messages the bottom window. Messaging is represented by the black line connecting the button to the window. On the right is an inspector giving the programmer the choice of legal messages to send the window.

ity of C with the object-oriented messaging facilities of Smalltalk–with a only few, easily learned extensions to ANSI C.

Unlike C++, Objective-C supports run-time binding, which allows the latest version of an object to be bound to the application at run time. Applications are not frozen in time when they are compiled. Thus as objects gain new functionality, applications using these objects automatically inherit all of their new features. For example, when NeXT extended its print panel to include faxing, all applications gained the ability to fax documents–even those commercial applications which had shipped one year earlier. Through run-time binding, NeXTstep gains complete extensibility of applications.

In addition, Objective-C's persistent objects provide debugging advantages over C++. In C++, objects only exist at compile time, and code is compiled into C. In Objective-C, objects exist at run time, allowing the developer to send messages to objects interactively in the same way that compiled code does. This is very useful for testing messaging schemes and the overall behavior of an object or group of objects.

The Application Kit. The Application Kit is NeXT's class library of optimized objects. It includes a set of core objects that provide the framework required by any application, and a powerful set of interface and support objects that provide advanced functionality well beyond the windowing toolkits found in other workstation platforms. In addition to standard interface objects such as scrollers, cursors, buttons, sliders, windows, and panels, the Application Kit provides objects for:

- Managing events
- Manipulating text data and editing multi-font text (including rulers and a complete spelling checker)
- Selecting colors
- Using standard dialogs (opening and saving files)
- Managing files
- Displaying TIFF and EPS graphics
- Faxing in Group III format
- Printing in PostScript
- Creating browsers

- Incorporating voice and sound (and a sound editor)

Everything in the Application Kit is fully extensible and all of its objects can be subclassed.

A Unified Imaging Model. Having co-developed Display PostScript with Adobe Systems, Inc., NeXT employs PostScript for all imaging, whether to screen, printer, or imagesetter. Developers need not worry about supporting two imaging models, one for screen display and one for printing. Using Display PostScript as a single imaging model ensures consistent results and allows developers to maintain only one set of graphics routines. NeXT works with the same Adobe Type 1 fonts that are widely available on devices ranging from desktop printers to imagesetters.

Support Tools. NeXTstep includes a range of additional tools: the *gdb* object-oriented debugger, MallocDebug (which measures an application's use of dynamic memory), PostScript previewing tools, a choice of UNIX text editors (Edit, a mouse-based programming editor; emacs; and vi), and AppInspector, which allows programmers to examine relationships between objects.

Operating System Integration. The NeXTstep development environment is well supported by NeXT's UNIX / Mach operating system.

Mach is NeXT's operating system kernel. Completely UNIX BSD 4.3 (Berkeley) compatible, it is the same kernel recently chosen for adoption by the Open Software Foundation (OSF). Mach offers an extremely efficient communications-oriented kernel, supporting multiple processors within computers and multiple threads within applications. The choice of Mach is consistent with NeXT's emphasis on object-oriented development because Mach is very efficient for messaging the objects within an application, for interapplications messaging, and for messaging across a network. Unlike other workstation programming environments, Mach and NeXTstep's programming tools were designed to provide an integrated platform for the development and use of object-oriented applications, optimized from the operating system on up.

In addition to full UNIX compatibility at the system level, NeXT supplies a complete set of familiar UNIX command line tools, such as *awk* and *sed*, as well as programming utilities like *gprof*.

VI. Developing With NeXTstep

What languages does NeXTstep support?

NeXTstep/Interface Builder's native language is Objective-C, based on ANSI C. Unlike C++, Objective-C is extremely easy for C programmers to learn, since it adds only a small set of extensions to standard C. Beyond ease of learning, NeXT chose Objective-C because of its support for dynamic binding. Dynamic binding allows developers to generalize programs so that they can handle a wide variety of object types without knowing all the possibilities ahead of time.

Third parties support many other languages including Pascal, Common Lisp, Ada, Eiffel, BASIC, FORTRAN, and COBOL. Absoft Corporation's object-oriented FORTRAN 77 offers access to standard UNIX tools and supports Interface Builder.

Developers can access applications written in languages that do not support Interface Builder directly. Through interapplications messaging, NeXTstep allows applications to be written in any language available on the NeXT platform. Custom Interface Builder programs can message applications written in any of these languages; to an end-user, it appears as if the program and the interface are one, seamless application.

Will existing C or C++ code have to be rewritten?

NeXTstep programs can utilize standard ANSI C, Objective-C, and C++. Included with NeXTstep is an Objective-C++ compiler. This compiler permits Objective-C and C++ to access objects written in either language; links C, Objective-C, and C++ into one application; and provides symbolic debugging of C, Objective-C, and C++ using NeXT's debugger. C++ and Objective-C are fully integrated into Objective-C++. Objective-C can call C++ methods, and C++ can call Objective-C methods.

While NeXT's own Application Kit will remain Objective-C based, organizations with a significant investment in C++ object classes can retain their investment while capitalizing on NeXTstep's powerful, integrated environment. Much of the core code of Lotus Improv was written in C++ before Lotus began its NeXTstep development. Thanks to Objective-C++, Improv was able to take advantage of Lotus' existing code base together with the added

productivity of NeXTstep. The result is the next generation of spreadsheets, developed first on the NeXT Computer.

Does NeXTstep support the standards needed in mixed computing environments?

Standard UNIX. NeXTstep is based on industry-standard Berkeley UNIX. Developers can compile and run UNIX code on a NeXT computer as easily as on any other computer that is based on UNIX 4.3BSD. NeXT plans to incorporate full POSIX compliance when BSD UNIX 4.4 is released.

Communications. Every NeXT computer supports thin coax and twisted-pair 10baseT Ethernet. NeXT's native support for TCP/IP and Sun's RPC (Remote Procedure Call) ensure that NeXTstep applications can support client/server computing. NeXT supports Sun's Network File System (NFS 4.0) as its own native file system, and its native mail system is based on industry-standard SMTP (Simple Mail Transfer Protocol). The third release of NeXTstep software will support Ethertalk and Novell client connectivity.

Through third-parties, NeXT computers can access many remote resources and tools, including AFS 2.0, Wang VS, 3270 and VT100/220 emulation, DECNET, X/Motif, Banyan, X.25, SLIP, PPP, and Macintosh file system support. Insignia Solutions provides full DOS emulation via SoftPC.

As interoperability and networking services and standards from such organizations as OSF are defined and gain market acceptance, NeXT will ensure their support within NeXTstep.

Database Connectivity. A variety of vendors offer SQL databases for NeXT, including ANSI-SQL compliant ORACLE DBMS and Sybase SQL Server. NeXT's Application Kit is being extended to include a powerful set of database objects so that programmers can access Sybase, Oracle, and other SQL databases from within Interface Builder.

Data and File Standards. Native file formats supported by NeXT include Encapsulated PostScript (EPS) and TIFF for graphics, and Microsoft's Rich Text (RTF) format for text. The NeXT file system also reads and writes DOS files. And NeXT supports JPEG compression/decompression and the ISO 9660 and High Sierra CD-ROM formats.

Are NeXTstep applications portable?

Properly written C, Objective-C, and C++ applications which separate user interface code from core functionality are very portable. The porting task is essentially similar to porting well-modularized code across other graphical user interfaces such as Windows, Motif, or the Macintosh. Applications ported from these environments to NeXTstep, however, involve less work, because implementing a user interface in NeXTstep requires significantly less coding.

Can NeXTstep support client-server computing? Are NeXTstep applications scalable?

Through support for TCP/IP, SQL, and Sun's RPC, NeXTstep applications are client/server ready. Mach's messaging facility provides very fast interprocess communications. Through RPC, NeXTstep applications can support cooperating processes on networked machines, from local UNIX-based servers to remote Cray supercomputers. NeXTstep's messaging facilities provide an ideal mechanism for integrating easy-to-use interfaces running locally with compute-intensive code running on remote cycle-servers, all within an easy-to-maintain NeXTstep application framework.

What are the benefits for end-users of NeXTstep applications?

Faster Time to Market. Organizations relying on custom applications as their competitive edge gain substantial benefit if those applications can be developed and deployed more quickly. In reducing a new application's time to market from a year or longer to a few months, an investment in NeXTstep and NeXT computers pays for itself quickly–often within a month after an application has been implemented.

> *"NeXTstep is a phenomenally productive environment for proprietary applications development. We were able to develop a fully-functional, fixed-income trader's workstation in less than three months. In other environments, it would have taken two, three, or even four times as long."*
>
> *– Robert M. Wilen, Vice President, SBC / OC Services L.P.*

Professional-Quality Interfaces. Programmers can quickly create applications with friendly interfaces that are as professional as shrink-wrapped applications from the major commercial software houses.

NeXTstep Applications Are Easier to Use. NeXTstep makes it easy to create user-friendly graphical user interfaces. And unlike other UNIX interface development tools, NeXTstep's Application Kit and Interface Builder encourage developers to create consistent user interfaces based on a common framework. When users have learned one NeXTstep application, they have learned the key elements of most applications: window and menu selection; file system navigation; font and color selection; cutting, copying and pasting between applications; and printing and faxing.

Integration. Being object-oriented and living in an object-oriented environment, NeXTstep applications are not standalone programs. All properly designed NeXTstep applications can cut, copy, and paste data from other applications, and request services from other applications. Any application can, for example, request services from *Webster's Dictionary* or NeXT's Digital Librarian, or mail documents to other users. More sophisticated applications can access remote databases, access mathematical models in *Mathematica*, or send numerical information to Lotus Improv for charting or analysis. Any custom NeXTstep application can avail itself of the facilities of many other NeXTstep applications, including the services of many commercially available tools. In this way, users can customize their working environment and create an integrated set of tools, where each new application adds value to all other tools present on the desktop.

VII. Summary: The NeXTstep Advantage

NeXTstep programmers receive all of the benefits of object-oriented programming:

- Reduced development time
- Decreased maintenance costs
- Reusability of code
- More robust and flexible applications

But NeXTstep is far more than a simple object-oriented language or toolkit. Beyond the general benefits of object-oriented languages, NeXTstep programmers–and only NeXTstep programmers–benefit from:

- A complete and integrated object-oriented development platform requiring programmers to write less code
- Interface Builder, which allows developers to assemble and test user interfaces and other application components in less time than with traditional methods and tools
- A complete user interface and applications framework which includes multifont text, printing, faxing, and file management
- A platform which allows completed applications to work together
- A platform whose interface tools encourage consistency, which leads to reduced training and learning time and greater satisfaction for end-users

VIII. Next Steps

Technology Decisionmakers' Seminar

For more information on how NeXTstep can be put to work in your organization, inquire about attending NeXT's one-day introduction to NeXTstep, the Technology Decisionmakers' Seminar.

Developer Class

To learn more about programming the NeXT computer, NeXT offers Developer Class, a week-long course in developing object-oriented applications using NeXTstep.

For information on attending NeXT's Technology Decisionmakers' Seminar or Developer Class, call **1-800 848-NeXT.**

Additional Reading

Budd, Timothy, *An Introduction to Object-Oriented Programming,* Addison-Wesley, 1991.

Cox, Brad, *Object-Oriented Programming: An Evolutionary Approach,* Addison-Wesley, 1991.

Lozinski, Christopher, "Why I Need Objective-C," *Journal of Object-Oriented Programming,* September 1991, pp. 21-28.

NeXT and Open Systems Standards, NeXT Computer, Inc., 1991.

NeXT Technical Documentation, Addison-Wesley, 1991.

The NeXTstep Advantage, NeXT Computer, Inc., 1991.

Weiner, Richard S. and Pinson, Lewis J., *Objective-C: Object-Oriented Programming Techniques*, Addison-Wesley, 1991.